Appendix G

Appendix G

POST Tools High-Level Architecture

Logical View Report

Unified Modeling Language Syntax

Attributes And Operations Excluded

Includes Documentation

TABLE OF CONTENTS

LOGICAL VIEW REPORT

Logical View

This is the high-level architecture for the POST Tools system. The logical view of the high-level architecture is the conceptual view, employing the SEI ABD decomposition representation of design elements, implementations, templates, requirements, decisions, and constraints.

Architecture

The Architecture package consists of the functional subsystem decomposition in design elements and implementations, the templates corresponding to each subsystem, and the deployment and concurrency views of the architecture.

Subsystems

The conceptual architecture subsystems are design elements and design element branches representing successive decompositions in a tree. These design elements are decomposed by functionality, quality scenarios, and constraints.

Browser Services

The user's desktop PC browser implements the save, print and view report functionalities. This includes the capabilities provided by browser plug-ins.

Database Query

An SQL database query and result set, using Java and JDBC/ODBC.

Hypertext Transfer Protocol

The system employs a standard HTTP interface for client-server communication.

Java Applets

This is a generic implementation class representating a Java applet. In our design the applet probably is implemented within the Silver Stream application server development and run-time environment.

Java Beans

This implementation suggests any Java component in a plug-in style component framework, such as a Java Bean or an Enterprise Java Bean.

Java Servlets

This implementation suggests any Java Servlet managed by an application server or web server. Java Server Pages implementations might also be suitable alternatives.

POST Tools

This class represents the top-level design element in our architecture, representing all functionality from Vision and stakeholder input, legacy system constraints, inter-project operation and dependencies, and other contributing factors. We decompose all ABD functionality from this design element.

Table Lookup

This is a generic implementation class suggesting that the system performs a table lookup to find a component given an index key.

Administration Services

The administration services package refers to the system administrator and POST field engineer activities in installing, configuring, maintaining, and otherwise supporting the hardware and software. This includes both the mobile components and the fixed components.

Account Management

This design element identifies the functionality required for managing user accounts and roles for administrator, field engineer, payload customer and mission operator access.

Account Tools

Implementations of the account management tools include the Windows/NT administration tools as well as the firewall and security tools.

Administration

The administration function encompasses installation, check-out, account management, system monitoring, updates, packaging, shipping, receiving, training, and other activities. We decompose this design element into administration for the USA location, administration for the customer field sites, and remote administration from the USA location to the field sites.

Backup and Restore

The backup and restore functionality includes writing file and configuration information to a tape for archival purposes as well as reading the archive in order to restore one or more files.

Data Table Backup

We employ a COTS product to implement the data table backup capability for the system, likely the table backup capability provided by SQLServer.

File System Backup

We employ a COTS file system backup product like ARCServe to implement this functionality.

Install Tool

We will use a COTS product such as InstallAnywhere or InstallShield to provide installation kit script and packaging capability.

Installation

Installation functionality includes: loading and configuring COTS products; loading and configuring POST application software; configuring the network addressing and hosts; creating accounts and privileges; generating digital certificates; configuring audits; performing installation verification.

Security Logs

This design element refers to security log functionality, tracing activity related to both approved and disappoved functions.

System Logs

This design element refers to system activity logs, tracking performance of activities on a local or remote system.

Windows NT Logs

We use Windows/NT to provide the necessary log functions. Sufficient capability is built-in to the operating system and is easy to administer.

Customer Site Services

The customer site services package refers to the functions available at the customer site, particularly referring to the POST tools PC server.

Customer Site Administration

Customer site administration functions include system logs, account management, backup and restore, and installation.

Remote Services

The remote services package encompasses the requirement that USA home users have network access to the field site, for the purpose of administration or development and test support.

Plug-In Install

The implementation of this function provides a way to insert, replace, or delete a Java-language plug-in within a services package.

Remote Administration

The remote administration design element encompasses the functionality for providing remote configuration and remote monitoring and control services to the system administrator. It can also provide limited remote monitoring and control services to a POST field engineer situated away from the customer site.

Remote Configuration

This design element identifies the functions that enable the administrator to perform field updates of DTDs, metadata files, component plug-ins, and corresponding mappings. This provides a way to add new capabilities after deployment.

Remote Monitoring

This design element identifies the remote monitoring and control functionality for interacting with a field system from the home location.

Toolkit Install

Toolkit installs include patch applications (systems and applications), certificate updates, automated install kit applications, and other pre-packaged services.

Update DTD

This design element provides a way to update the set of DTDs in the field. Update could mean replacement or addition. It may include updating metadata documents and plug-in mapping tables.

Update Plug-In

This design element provides a way to update a component plug-in tool in the field. This includes updating through the plug-in component framework as well as updating a mapping table of DTD or metadata specifications to plug-ins.

Update Tools

This design element provides a way to update a specific tool (POST or COTS) or the operating system on a system situated in the field. Updates include complete installations as well as patches.

XML File Install

The DTD files reside in a well-known location. This may also include or substitute resource description framework (RDF) files.

pcAnywhere

We employ a COTS product pcAnywhere to provide the remote monitoring and control capability.

USA Site Services

The USA site services package refers to the functions available on the home servers for administering the system, mission operator access, initialization, security, and other features.

Customer Initialization

This design element captures the functionality necessary for the system administrator to initialize the shared data repository for access by a new customer or payload. This includes applying the standard schema to database tables, providing any flat files, initializing configuration management information, creating accounts, and configuring the network.

DBA Tool

We employ the features of the COTS database product to perform much of the initialization activity. Remaining activities must be done manually by the system administrator.

Firewall Logs

We employ a COTS firewall product to provide these security logs.

Network Administrator Tools

We employ COTS products for the network administration including: firewall management tools; VPN management tools; analysis tools; OpenView; and other products. These are not tools that POST builds.

Network Management

The network management functions include set-up and tear-down of routing access for remote sites, as well as VPN security functions, certificate distribution and revocation, and intruder detection.

USA Site Administration

The USA site administration functions include system logs, security logs, account management, backup and restore, installation, customer initialization, and network management.

Configuration Management Services

The configuration management services package captures the usual CM functionality regarding version control, artifact access, and auditing.

Access Control

The access control functions provide management and use of data unit priviledges. The management interface is a GUI that provides controls for administrator changes to user privileges. The run-time interface in Verify Access verifies user credentials against these privileges and grants or denies access.

Audit

The configuration management auditing functions provide rule-based validation of data collected from the user. There are several levels of auditing, including but not limited to

field-level validation, field-to-field validation, and record validation. A user interface provides selection of comprehensive audits for some products, and provides access to report generation.

Configuration Management

The configuration management design element captures the functionality behind both user-centric and administrator-centric operations upon data units. Functionality includes security, auditing (quality), and version control.

Field Audit

Field audits provide field content verification and validation within the client-side tools. This includes some field-to-field validation where applicable.

Record Audit

The record audit function pertains to server-side verification and validation on entire records. This includes record-to-record auditing if appropriate.

Source Safe

Current plan is to use the Microsoft Visual Source Safe product to provide versioning, recovery, administrative, and API services.

Verify Access

Verifies that the user has write access priviledge for the data item being imported, exported, or otherwise accessed. The user should have the item checked out and locked before overwriting the item with the imported or collected data.

Version Control

The version control design element represents the user interface and functionality for version numbering and reporting, data unit check-out status, and data unit promotion.

Data Collection Services

The data collection service package refers to the functionality of the tools prompting the payload customer to provide information about the payload. This primarily encompasses the command and data definitions and a variety of mission operations information.

Data Collection

The data collection design element represents the functionality of data gathering by prompting the user for entries or by importing foreign files. Both methods require configuration management. The prompting method refers primarily to command and data information and mission operations information. A generic import approach enables migration of a variety of flat-file types into the POST system. Imports include legacy command data files; training models; ground displays; Cargo PC applications; and so on.

Import Data Services

The import data services provide the payload customer with a way to import foreign data and artifacts into the data collection subsystem. This activity can be performed in the field. Importability is determined by the availability of translators, and by the CM status of the data unit to be overwritten.

Import Data

The import data functionality is provided through a set of translator tuples of the form <DTD, plug-in>. Each element of the set provides the structure validation (in the DTD)

and the matching translator (plug-in) for POST-compatible formats. We decompose the import data functionality into several supporting design elements, including: a GUI to identify source files and structure definitions; translators to produce the desired content; and remote reconfiguration services for the translator set.

Manual Entry Services

The manual entry services represent the activities involved in soliciting information from the payload customer and capturing this information in the appropriate data units.

Manual Entry

The manual entry design element provides functionality which we decompose into configuration management, editing, validation and storage services. The interaction is browser-based client-server style conducted primarily via forms.

Editing Services

Manual entry editing services support interaction with the payload customer to elicit information for the POST-related data units. The client-server architecture style provides for a brower-based interaction with both client-side and server-side support of the functionality.

Edit Data

This design element represents the functionality of prompting and capturing information from the user in a friendly manner.

Modify Entries

This design element represents the functionality of changing information that is already partially complete, including features such as cut and paste or modifying a text field value.

Prompt Entries

Part of the editing functionality is to prompt the user for the desired information. We use primarily form-level prompting (with surrounding and controlling navigation services) and user-friendly and standard-semantic controls.

View Entries

In addition to prompting the user for information entry, the subsystem provides the expected ability to see what value already exists in a field. This design element provides the capability to view information that has already been entered without affecting that information.

Validation Services

Manual entry validation services provide the functionality necessary to perform quality inspections of the data the payload customer provides.

Client Side Validation

This design element performs the validation tasks that can be done at the client. These include field syntax and field-to-field validations.

Field Validation

Implementation of field validation is performed by Java programs. Errors are returned to user via client GUI, pre-empting submission to the server if possible.

Record Validation

Implementation of record validation is done with Enterprise Java Beans, performing comprehensive validation on and across records. The EJB returns error results to the client.

Server Side Validation

This design element performs the validations that must be done on the server. These include record and record-to-record validations.

Validate Data

The validate data design element captures two perspectives on manual entry validation strategies. Some validation can be done on the client, within the tools and forms presented to the user. The remaining validation must be done on the server, after the user submits his entries, for comprehensive validation.

Storage Services

The data collection storage services provide the functionality for accessing databases and file systems.

Data Access

This design element represents the functional capability to read or write data from a database or from the visible file system.

Database Read

This implementation of a database read represents a read from the working data store (WDS) using SQL queries.

Database Write

The implementation of a database write represents a write to the working data store (WDS) using SQL queries.

File System Read

This implementation element represents a read function or utility in the local file system.

File System Write

This implementation represents a standard utility or function for writing to the local file system.

Data Consumption Services

The data consumption service package refers to functionality that delivers processed payload information to a user in the form of reports or queries or products.

Data Consumption

This design element contains the functionality for all users to consume the data and products captured by the POST tools. Consumption includes exports, reports, deliveries, and other uses.

Export Data Services

The export data services provide the payload customer with a way to export foreign data and artifacts from the data collection subsystem. This activity can be performed in the field. Importability is determined by the availability of translators, and by the CM status of the data unit to be overwritten.

Export Data

This design element provides the functionality to move a POST-native data unit into some foreign format. We employ the use of plug-ins to enable future additions.

Product Services

The product services package contains the functions necessary to produce a given product, including data gathering and number crunching.

Generate Products

Contains the functionality to issue a product generation request from the client, generate the product on the server, and have a summary of results sent back to the client for analysis.

Identify Product

Provides the functionality necessary for the user to select a desired product from a list of installed product types.

Summarize Results

Sends a summary of the product processing activity from the server to the client. We assume this is done via HTTP using the browser-server connection.

Processing Services

Contains the functionality to issue a product request from the client, generate the product on the server, and have the summary of results sent back to the client.

Create Product File

This function uses the incoming data and a format specification to produce the product file. The functionality is provided by a component plug-in for each product type.

Execute Product Engine

The product production engine gathers the required information and produces a file of the appropriate type. There are no constraints on the file type other than types produceable by the production engine plug-ins and translatable into reports or consumed by outside customers.

Gather Product Data

Gathers the data necessary for the selected product. Data can come from database table queries or from flat files accessible from the server.

Gather Product File

Gathers product data from a flat file. By flat file we mean any data not coming from a database server query.

Gather Product Records

Gathers the necessary product data using a database server query.

Map Product Engine

This function maps the selected product identifier to its corresponding production engine. The engine is a component plug-in that knows how to produce the product. The engine runs on the server.

Produce Product

Provides the functionality to identify on the client a product from a pre-defined list, activate the corresponding production engine on the server, and send the summary output back to the client.

Product Engine Lookup

Uses the lookup services to find the production engine given the product identifier.

Derived from Table Lookup

Production Component

This implementation is the component plug-in that produces the product. It is an instantiation of an EJB.

Product Consumption

Provides a summary of the product processing activity.

Consume Product

This functionality represents the things that the user can do with the product output from the client browser. We assume that the product cannot be viewed by a plug-in, so we delegate interactive processing to the report generation subsystem. Instead, we merely provide here a summary of product analysis.

Report Services

The report services package provides the functions necessary to produce various reports from the current data content.

Generate Reports

Contains the functionality to issue a report request from the client, generate the report on the server, and have the result sent back to the client for further processing.

Identify Report

Provides the functionality necessary for the user to select a desired report from a list of installed report types.

Transfer to Client

Sends the report output file from the server to the client. We assume this is done via HTTP using the browser-server connection.

Production Services

The production services package contains the functions necessary to produce a given report, including data gathering and rendering.

Execute Report Engine

The report production engine gathers the required information and produces a file of the appropriate type. There are no constraints on the file type other than types produceable by the rendering engine plug-ins and consumable by the browser features and plug-ins. File candidates include DOC, PDF, PS, RTF, HTML, XML, TXT and so on. Preferred types are PDF, RTF, HTML and TXT.

Gather Report Data

Gathers the data necessary for the selected report. Data can come from database table queries or from flat files accessible from the server.

Gather Report File

Gathers report data from a flat file. By flat file we mean any data not coming from a database server query.

Gather Report Records

Gathers the necessary report data using a database server query.

Map Report Engine

This function maps the selected report identifier to its corresponding formatting or rendering engine. The engine is a component plug-in that knows how to produce the report. The engine runs on the server.

Produce Report

Provides the functionality to identify on the client a report from a pre-defined list, activate the corresponding formatting and rendering engine on the server, send the output back to the client, and view the results using client tools.

Render Report Data

This function uses the incoming data and a format specification to render the output document. The functionality is provided by a component plug-in for each report type.

Rendering Component

This implementation is the component plug-in that produces the report. It is an instantiation of an EJB.

Derived from Java Beans

Report Engine Lookup

Uses the lookup services to find the rendering engine given the report identifier.

Derived from Map Report Engine, Table Lookup

Report Consumption Services

The report consumption package contains the functions that the user employs to interact with a completed report file.

Consume Report

This functionality represents the things that the user can do with the report output from the client browser. The browser may be able to handle the report output file directly, or it may employ a plug-in. The browser support will include the capability to view, print or save the file, and we assume that the system need not provide special handling of these files.

Print Report

The user will be able to send a report file to a printer.

Save Report

The user will be able to store a report file to his local file system.

View Report

The user will be able to view (but not edit) a report file on his screen.

Development and Test Services

The development and test services package contains the functionality of the system when the payload customer is developing and testing certain products. The development aspect of the package refers to the training model, cargo PC application software, or displays. The test aspect refers to the tool support of the customer testing these developed products, especially with unit testing capabilities or integrated testing capabilities using the orbiter-in-a-box.

Development and Test

The development and test design element captures a broad range of functionalities for the payload customer. The payload customer can develop ground and Cargo PC displays and develop training models. He can also perform unit testing and integrated testing on these products. Integrated testing supports use of the orbiter-in-a-box, Cargo PC, and payload.

Development Services

The development services package contains support for payload customer development of certain deliverable products.

Development

This design element provides the functionality for developing certain products in the field, using the tools we provide in the POST tools platform.

Display Development Services

The display development services package contains those functions associated with the payload customer's development ground, instructor, and Cargo PC displays.

Display Development

The display development design element provides the functionality for developing the platform-portable displays. We decompose this design element's functionality into display manipulation and display library design elements.

Display Library

The display library design element provides a set of GUI components that can be assembed into a component framework. Components unique to POST tools include: button group, custom time, dynamic object, gauge, linear meter, object icon, plot, slider, text, text symbol, command.

Display Manipulation

The display manipulation design element provides the editor for selecting and laying out graphical components onto a display.

Portable Display Builder

The portable display builder (PDB) provides an integrated development and run-time tool using a collection of Java beans to support telemetry monitoring and commanding displays.

Training Development Services

The training model development services package contains those functions associated with the payload customer's development of the payload training model.

Executable Generation

The executable generation design element functionality builds an executable program from the synthesized payload training model C language program source code.

Model Build Tool

The executable generation implementation uses the Microsoft Visual C++ compiler to compile synthesized C source code and link ISP client and reflective memory network libraries into an executable file. The implementation also supports unit test capability using the embedded simulation capabilities on the POST tools PC.

Model Creation

The model creation design element provides an interactive GUI through which the payload customer specifies the logic for the payload training model.

Model Edit Tool

The model edit capability implementation uses the MATRIXx Xmath and SystemBuild COTS products on the POST Tools PC.

Model Synthesis Tool

The model synthesis implementation uses the MATRIXx Xmath and AutoCode COTS products on the POST Tools PC.

Source Code Synthesis

The source code synthesis design element translates the model logic file into C source code that can be compiled and linked with support libraries.

Training Model Development

This design element captures the high-level functionality associated with creating the payload training model. We decompose the design element into functions for model creation, source code synthesis, and executable file generation.

Reconfiguration Services

The reconfiguration services package contains support for modifying data files for test activities after development activities have changed the content.

Data Files

The data files design element represents the collective functionality of the data files supporting development and test activities. These data files include the ISP dictionary,

SMS data stores, OI mass memory image, Shuttle data tape, and GAF simulation configuration files.

GAF Config

The GPC emulator employs GAF files to configure itself at startup. For example, a GAF file specifies the configuration of MDM channelization.

ISP Dictionary

The ISP dictionary provides the list of symbols, characteristics, and nomenclature for the ISP server.

ISP Null Server NT

This entity represents the null implementation of an ISP server on Windows/NT.

ISP SITF Server NT

This entity represents the SITF data source implementation of an ISP server on Windows/NT.

MCDS Program

This entity is the Java-language MCDS emulator program for interacting with the orbiter flight software running in the GPC emulator.

OI Mass Memory

This entity is the orbiter flight software mass memory image.

OIAB Programs

The orbiter-in-a-box programs include a collection of tasks to run the GPC emulator components, SMS models, and ISP server on the VxWorks operating system.

PDB Beans

The portable display builder graphical components and data provider beans are in a Java archive file.

PDB Program

The portable display builder program is the Java-language development and run-time container for PDB beans.

Reconfiguration

The reconfiguration design element represents the functionality necessary to support development-motivated changes to software and data files in the field, especially prior to an integrated test activity. We decompose this design element into design elements for reconfiguring software and reconfiguring data files.

Reconfigure Data

The reconfigure data design element provides for the field-update of development and test data. The functionality includes an administration interface for determining file access permissions.

Reconfigure Software

The reconfigure software design element provides for the field-update of development and test programs. The functionality includes an administration interface for determining file access permissions.

SMS Data Stores

This entity represents the SMS data store images that the orbiter-in-a-box can use to start the flight software and SMS models in a consistent state.

Shuttle Data Tape

This entity represents the channelization specification portion of the Shuttle Data Tape content.

Software Files

The software files design element contains the functionality for configuring software tools in the POST tool set.

Test Services

The test services package contains support for unit testing and integrated testing of the customer's products using high-fidelity simulations.

Test

The test design element provides the functionality to test products at the customer's site. We decompose this design element into unit test and integrated test design elements.

Integrated Test Services

The integrated test services package contains support for comprehensive high-fidelity testing of a customer's product in concert with one or more participating elements.

Integrated Test

Provides functionality to perform testing of the payload training model, displays, and Cargo PC at the customer's facility. We decompose this design element into integrated test design elements for Cargo PC, displays, training models, and payload.

Cargo PC Test Services

The Cargo PC test services package contains support for integrated testing with the orbiter-in-a-box (to test command and data products), and either the training model of the payload or the real payload.

Cargo PC

This is the physical Cargo PC, including its system software, application software, and communication links.

Cargo PC Displays

This design element represents the Cargo PC displays that the payload customer produces in the field. These displays show orbiter and payload telemetry, computed values, and other information. These displays also provide a commanding interface for the crew. Because these are platform-neutral displays they may be hosted on other machines.

Cargo PC Integrated Test

The Cargo PC integrated test design element captures the POST tools provisions for testing the Cargo PC application software and GPC payload command filter (GPCF) data structures. We decompose this integrated test design element into Cargo PC test, orbiter simulation, payload simulation, and payload system functions.

Cargo PC Test

This design element provides the functionality to support testing of the astronaut crew interaction with the Cargo PC. This function requires the Cargo PC itself (provided from another project) and the Cargo PC system and application displays.

Display Integrated Test Services

The display integrated test services package contains support for payload customer testing of various displays in concert with the payload, payload training model, or Cargo PC.

Display Integrated Test

The display integrated test design element provides the functionality to test customer-generated displays against an orbiter simulation. We decompose this functionality into ground display test, instructor display test, Cargo PC test, and orbiter simulation design elements.

Ground Display Test

This design element supports check-out of the customer's instructor displays for controlling the payload training model. In this integrated test subsystem the data source is the ISP server running in the orbiter-in-a-box platform.

Ground Displays

This represents the Java and ISPresso ground application displays that the customer creates with the PDB tool.

Instructor Display Test

The instructor display test design element provides the functionality to test the customer-generated displays that are designed to control the payload training model.

Payload Instructor Displays

This represents the payload instructor displays, written with PDB and ISPresso, that enable the instructor to control the behavior of the payload training model.

SMS Instructor Displays

This implementation represents the Java displays that the NGFCT collection of instructor displays employs to control the SMS models.

Orbiter Simulation Services

The orbiter simulation services package contains support for high-fidelity simulation of certain orbiter avionics components in the field. This simulation is necessary to support integrated testing, especially with the Cargo PC.

Cargo PC Commands

The Cargo PC commands design element provides the functionality for supporting a GPCF-controlled commanding process between the orbiter and the Cargo PC.

Cargo PC Telemetry

The Cargo PC telemetry design element provides the functionality for supporting a PCMMU-generated telemetry stream between the orbiter and the Cargo PC.

GPCF Simulator

The GPCF simulator implementation is a composition of functions provided within the orbiter-in-a-box. The GPC emulator runs the SM flight software, which includes the GPC payload command filter (GPCF). The GPCF issues instructions on an MDM channel, which the GPC emulator models through a codec and MDM interface hardware. These commands go to the Cargo PC MDM interface, which the Cargo PC system software manages.

MDM Simulation

The MDM simulator in the orbiter-in-a-box provides support for discrete input and output and analog input signals. The card type support includes DOL, DOH, DIL, DIH, and AID. The MDM interface card can provide SIO support if needed.

Orbiter Simulation

The orbiter simulation design element represents the functionality necessary to drive the data interfaces with the payload and the Cargo PC. Each device has a pair of simulated interfaces: one for telemetry and one for command.

Orbiter-in-a-Box

This implementation element depicts the orbiter-in-a-box tool as an aggregation of implementations for each of the supported subsystem interfaces. The orbiter-in-a-box consists of simulations of the PCMMU, GPCF, PDI, MDM and PSP. In addition, the orbiter-in-a-box provides an MCC front-end simulation with an embedded ISP server.

PCMMU Simulator

The PCMMU simulator in the orbiter-in-a-box generates the complete telemetry stream for output to the Cargo PC. The GPC emulator, PCMMU emulator, and SMS models produce the stream content, writing the data to a PCM signal generator for output to the Cargo PC.

PDI Simulator

The PDI simulator in the orbiter-in-a-box reads the complete telemetry stream from the payload. The GPC emulator, PDI emulator, and PCMMU emulator read the incoming data and incorporate it into the composite telemetry stream. The orbiter-in-a-box implementation employs a bit synchronizer and decommutator to read the incoming stream.

PSP Simulation

The PSP simulator in the orbiter-in-a-box generates the command stream and signal outbound to the payload. The GPC emulator, MDM emulators, and PSP model produce this signal content. The orbiter-in-a-box implementation employs a PCM generator and phase modulator to generate the electrical signal.

Payload Commands

The payload command design element represents the functionality of the orbiter subsystems generating commands for the payload through the orbiter data communication subsystems. We implement this functionality in the payload signal processor (PSP) and multiplexer-demultiplexer (MDM) subsystems.

Payload Telemetry

The payload telemetry design element represents the functionality of the payload producing a data stream through the orbiter data communication subsystems. We implement this functionality in the payload data interleaver (PDI) and multiplexer-demultiplexer (MDM) subsystems.

Payload Integrated Test Services

The payload integrated test services package contains support for including the customer's payload in integrated tests with the Cargo PC or various displays.

Payload

This implementation class represents the customer's payload.

Payload Integrated Test

The payload integrated test design element represents the test activities that include the customer's payload within and integrated test session. The POST tools project does not provide specific payload testing capabilities, but it does provide the ability to include the payload in a test of Cargo PC software or displays.

Payload System

The payload system design element represents the data communication interfaces with the orbiter, in particular the PSP, PDI and MDM.

Training Model Test Services

The training model test services package contains support for integrated testing of the training model in concert with the Cargo PC.

Orbiter Simulation Controls

This design element provides the functions for managing the orbiter model side of the simulation.

Payload Model

This is the executable model of the payload from the development subsystem's model build design element.

Payload Simulation

Provides the functionality to simulate the behavior of the payload, communicating with other system test components as the model would when deployed in the SMS.

Payload Simulation Controls

This design element provides the functions for managing the payload model side of the simulation.

Training Model Integrated Test

Supports an integrated test of the payload training model at the customer's facility. Integration includes single-user support for running the training model, instructor displays, payload simulator service, and Cargo PC simultaneously.

Payload Server Services

The payload server services package contains representations of the SMS interfaces and services necessary to support development and test of the payload training model.

Payload Server

The payload server design element is responsible for the functionality of moving data to and from the payload model and orbiter model data pools.

GPCE Model Services

The GPCE model services package contains support for implementation of the avionics interfaces necessary to support an integrated test.

GPCE Models

These are the orbiter avionics models implemented in the GPC emulator platform for NGFCT, modified if necessary to suit implementation within OiaB.

MDM Adapter

Communication adapter to make communication scheme transparent to MDM model. Implemented in C++ with GAF reconfiguration files.

MDM Model

This element provides the functionality of the orbiter payload MDM model, communicating discretes, analogs and serial I/O with the payload model. The implementation employs and adapter to enable the MDM model to communicate with the payload, via hardware, or the payload model, via software.

PDI Adapter

Configuration adapter to make communication scheme transparent to PDI model. Implemented in C++ with GAF configuration files.

PDI Model

This is the orbiter payload data interleaver model, incorporating telemetry received from the payload model into the orbiter PCMMU model. This implementation uses and adapter enabling the PDI model to use either a hardware or software input.

PSP Adapter

Configuration adapter to make communication scheme transparent to model. Implemented in C++. Uses GAF for reconfiguration, if necessary.

PSP Model

This is the orbiter payload signal processor model providing the functionality of sending commands to the payload model. This implementation uses an adapter to make the PSP model use either a hardware or software output.

Mode and Control Services

The model and control services package contains support for user or instructor interaction with the integrated test simulation, enabling control of the session.

Generate Data Store

Creates a snapshot of the payload model data pool and writes it to a file system for later recovery.

Mode and Control

This element is responsible for the control of the simulation models, including instructions for data store saving and recovery, initialization, and moding.

Payload Data Read

Given a source file name, reads the file content and loads payload data pool memory. Specific implementation is an issue associated with plug-and-play server.

Payload Data Save

Given a target file name, reads payload data from the data pool and writes the content to a file. Specific implementation is an issue associated with plug-and-play server.

Payload Server Displays

This represents the GUI displays, built with Java and ISPresso, that the instructor or test engineer uses to control the payload server during an integrated simulation.

Recover Data Store

Reads a file from the system and loads its contents into the payload data pool, recovering a model state from a previously saved file.

Runtime Controls

Provides a graphical user interface through which the instructor (or test engineer) can start, stop, and mode a simulation, or generate and recover a data store.

Model Communication Services

The model communication services package contains support for the data exchange between the user, the orbiter simulation, and the payload simulation.

Instructor Display Ops

Provides services for the instructor to communicate with the payload models and SMS models via instructor controls pages.

Payload Model Comm

This element is responsible for implementing and servicing the communication between the payload training model and the payload server. At the payload server, other functions integrate the payload model and SMS model data streams. This is also responsible for the ISP communication between the payload instructor displays and the payload training model.

Reflective Memory

Provides reflective memory network services, including communications protocols and device drivers.

ScramNet

Reflective memory network implementation is Systran SCRAMNet+.

SMS Model Services

The SMS model services package contains support for the orbiter models running in the orbiter-in-a-box.

Data Gather

Reads model term values from the SMS data pools and writes it to the reflective memory interface to the payload model. Data of this type includes electrical power, thermal, environment, etc.

Data Glue

Data glue code that pastes together the data pool and reflective memory interface. Potential reuse with payload server libraries. C or C++.

Data Scatter

Pulls data from the payload model reflective memory interface and writes it directly into the SMS data pool term locations.

ISP Server

This implementation is a standard ISP server, running either on the POST tools server machine or on the orbiter-in-a-box machine.

SMS Models

This design element represents the SMS training models of the environment and the orbiter.

SSP Display

This implementation suggests a Java GUI representing the standard switch panel.

Switch Panel

This functionality represents the need to provide a GUI of the standard switch panel and use it to set data term values in the SMS data pools.

Server Reconfiguration Services

The server reconfiguration services package contains support for initializing the payload server implementation running in the orbiter-in-a-box.

Data Pool Lookup

Provides a "find term" service that returns an address for a given symbol identifier.

Data Pool Map

Maps term identifiers to data pool addresses so that the payload model can subscribe to values from the SMS models.

Import CDT

Provides a way to acquire payload configuration information from the CDT and to configure the PSP and PDI models.

Import SDT

Provides a way to acquire MDM channelization information from the SDT, and configure the MDM models.

MDM Channelization

Provides MDM element channelization from the Shuttle Data Tape.

Payload Data Config

Provides PSP, PDI and other payload information as if it were from the payload data tape (but uses CDT in the field).

Payload Server Reconfig

Provides the functionality to reconfigure the payload server according to telemetry definitions and other items that the payload customer can reconfigure.

Unit Test Services

The unit test services package contains support for preliminary but comprehensive testing of a product in a standalone configuration.

Display Unit Test

This design element supports the unit testing of ground displays and Cargo PC displays using a scripted telemetry server as a data source.

ISP SITF Server

This is an ISP Source-Independent Telemetry File (SITF) server running on the POST Tools PC. The user will have to generate a SITF by hand, using the format specified in the user's guide. The user will start the SITF server using the SITF as a telemetry source. The PDB client will connect to this server using standard ISP. The user's SITF script can contain malfunctions and such to test functionality.

Training Model Unit Test

This design element represents the required functionality to support unit test of the customer's payload training model.

Unit Test

The unit test design element provides functionality to perform unit testing for a customer-developed product. We decompose this design element into unit test support for training models and for displays.

Navigation Services

The navigation services package contains the functionality for payload customer navigation through the tools software, especially dialogs, menus, shortcuts, and wizards.

Navigation

This design element provides functionality for navigating among the services and pages presented by the system, as well as the files available to the user but not provided by the system.

External Navigation Services

The external navigation services package contains functions provided by components outside of the POST tools software, particularly file system access.

External Navigation Subsystem

External navigation elements refer to file selections and invocations on the client PC or LAN, particularly files that are not directly part of the POST tools system.

Identify File Services

The identify file services package contains functions required for the user to identify files on the file system. There are both "general" and "constrained" idenfitication services.

DTD Browser

This is a standard file selection dialog, however we use it to identify to the system which DTD applies to a particular activity.

File System Browser

This implementation refers to a standard file system browser dialog box, such as that provided by the Java Foundation Classes. The box should include a directory browser, type filter, file selection, and the other usual capabilities.

Derived from Source File Browser, DTD Browser

Identify File

This design element requires the user to identify which file is the source or target for a read or write activity.

Select DTD

Once the user identifies the DTD file in the file system browser, he "selects" or activates the subsequent process from the browser. We assume we can assign hooks to be activated by this open function to transition to the next activity.

Select File

Once the user identifies the file in the file system browser, he "selects" or activates a subsequent process from the browser. We assume we can assign hooks to be activated by this open function to transition to the next activity.

Select File Hook

This implementation provides a method hook into the file system selection so that file "open" invocation activates a subsequent step in the process.

Source File Browser

This function provides a typical file system browser that the user employs to identify the source file for the input. Features can and should include directory changes, filters, and so on.

Internal Navigation Services

The internal navigation subsystem services package contains functions provided by components inside the POST tools software, particularly browser controls.

Checklist

The checklist provides a task-oriented guide for the user to follow, guiding him through a series of pages or links associated with a selected task.

Form Manager

The form manger controls content presentation and navigation for a single task. This implementation is a Java class.

Internal Navigation Subsystem

Internal navigation elements refer to the user's manual navigation among the pages and tools that the system provides. Examples include web page traversal, desktop shortcuts to tools, and form tabs.

Product List

This is a list of the possible products that the server can produce. The list contains the items from a remotely-configurable metadata file that identifies the products and the component plug-in that can produce the product.

Report List

This is a list of the possible reports that the server can produce. The list contains the items from a remotely-configurable metadata file that identifies the reports and the component plug-in that can produce the report.

Shortcuts

The implementation of this feature is provided by web browser bookmarks or desktop shortcuts.

Site Map

This is a composite view of all of the links that can be reached through the system services. The links should be categorized by major function.

Task Manager

This function provides a lower-level control over the selection and presentation of pages associated with a particular step in a multi-step task.

Tool Selection

If the user knows which tool he wants to run, then we let him jump straight to that tool. This contrasts with the task sequence service.

Way Points

This function enables the user to bookmark a certain page or location for subsequent recall.

Web Page

This implementation refers to a web page containing HTML or Javascript.

Translation Services

The translation services produce a POST-compatible data unit to or from an foreign source file whose structure has been validated against a DTD. A translator function for the DTD-validated parse tree produces the desired data unit.

CDT Tables

The translator produces CDT database tables if appropriate.

Execute Translator

Once we have the parsed object tree and a handle on the translator function, we execute the translator to traverse the tree and produce the desired data representation.

MOT Files

The translator produces MOT files if appropriate.

MOT Tables

The translator produces MOT database tables if appropriate.

Map Translator

This functionality maps the chosen DTD to a specific built-in (plug-in) translator for that document type. The idea is that we can look-up in some table the appropriate translator using the DTD as the key.

Translation

This function translates the output of the XML parser into a native (local) friendly format which can be stored in the POST file system or data tables, or translates a POST file system data unit into a foreign system format.

Translator Lookup

The translator lookup implementation uses the lookup services to find a translator given the DTD.

Derived from Table Lookup

Validate Structure Services

The import data subsystem requires a way to validate whether a foreign file can be imported. The validate structure services package contains the necessary functionality to confirm compatibility and load the foreign file into the POST system.

Parse Source File

Given the import source file and its corresponding DTD, the parsing tool will create a parse object tree and a validity flag. If the process produces an invalid structure indication, the system will not import the file.

Read DTD File

Provides functions to read the desired DTD file from the user-specified location. The user matches the import source file to a POST DTD.

Read Source File

This element provides functions to read the import source file from the user-specified location. The source file is an XML file that can have any sort of tag and structure recognized by a POST DTD.

Validate Structure

This element performs a validation of the source file. It performs the validation according to a DTD specified by the user or by mapping a file to a prespecified DTD.

XML to Java Tool

This implementation element represents a typical XML to Java parsing tool, such as the IBM xml2java package. The result is a valid parse tree with known objects at the nodes.

Templates

The conceptual architecture templates capture design element implications on the deployed system.

Adapter

An Adapter class implements an interface known to its clients and provides access to an instance of a class not known to its clients. An adapter object provides the functionality promised by an interface without having to assume what class is being used to implement that interface. We use adapter classes within orbiter-in-a-box to enable the simulation ports to employ either software or hardware interfaces.

Balking

The Balking synchronization pattern models the behavior of one object aborting the message transfer if the other is not immediately available.

CM

The Configuration Management API provides for check-in, check-out and other management features that the CM system implements. This likely uses the OLE interface to the Microsoft VSS product.

DOM

The document object model for XML provides basic programming capabilities for random, read-write processing XML documents including creation of the object document and node tree.

Dynamic Linkage

The dynamic linkage pattern allows a program, upon request, to load and use arbitrary classes that implement a known interface. This pattern is useful for the plug-in model of report generation and import/export translators. The likely implementation will include the Virtual Proxy pattern as a class loader.

Enterprise JavaBeans

The Enterprise JavaBeans API provides a component specification and event handling mechanism for session-specific pluggable objects on servers. For example, the report generation capability uses EJB to implement plug-in report generators.

JAXP

The Sun Microsystems Java API for XML Parsing (JAXP) enables basic functionality for reading, manipulating, and generating XML documents through pure Java API's. This might be considered a container of the DOM and SAX API's and including the Java Project X parser.

JavaBeans

The JavaBeans API provides a component specification and event handling mechanism for pluggable objects. PDB uses JavaBeans for its graphical objects. The import/export tools use JavaBeans for plug-in capability.

Mediator

The Mediator pattern uses an object to coordinate state changes between other objects. Putting the logic in one object to manage state changes of other objects, instead of distributing the logic over the other objects, results in a more cohesive implementation of the logic and decreased coupling between the other objects. We employ a version of this pattern in the orbiter-in-a-box tool for process management.

SAX

The simple API for XML provides basic programming capabilities for serial, stateless, processing XML documents.

Strategy

The Strategy pattern encapsulates related algorithms in classes that are subclasses of a common superclass. This allows the selection of algorithm to vary by object and also allows it to vary over time. This pattern will be useful in the user-guidance and wizardry implementations.

Virtual Proxy

The Virtual Proxy pattern hides the fact that an object may not yet exist from its clients, by having them access the object indirectly through a proxy object that implements the same interface as the object that may not exist. We employ this technique as a class loader for user-selected actions.

Constraints

The constraints package captures design decisions that are pre-specified. The constraints, along with the business goals, affect the derivation of design decisions. Included are legacy systems and interfaces, COTS or implementation effects, etc.

Model Fidelity

The development and test capabilities providing orbiter-like simulations must have the highest quality models available. Cycle timing must match orbiter specifications.

Payload Cabling

The development and test portion of the hardware architecture must support flight-like cabling for the payload. Prior decisions have specified that this cabling be in the form of the SMCH.

Plug-and-Play Payload Model

In order to support the customer's development of the payload training model, the development and test environment must support a communications interface that appears to be the SMS payload server. The existing ICD and engineering done to support the plug-and-play environment in the SMS applies to the implementation of the payload server on the orbiter-in-a-box tool.

Portable Platform

The tools hardware must be portable to the customer's facility. The Vision document identifies portability as baggage transportable by commercial airlines.

Decisions

The decisions package captures the architecture design decisions and issues. The decisions reflect outcomes of deliberation with regard to functional decomposition, implementation choices, architecture style, utility, quality and operation.

Accuracy

Accuracy decisions and issues concern the architecture considerations for fidelity, depth and breadth of the development and test capabilities.

MDM Output Signals

The MDM output signals from the Acromag IP mezzanine modules must be conditioned to match orbiter specifications. The cards are optically isolated. Need to design conditioning for overvolt protection, fault current limits, fault volt emission, power-ground isolation, and power-off impedance.

PSP Output Signal

The signal output of the SBS-4416VF card, with PSK modulation enabled, is in the hundreds of millivolt range instead of the 2-3 V range provided by the orbiter PSP. We will need to condition this output signal to provide a better match to the orbiter-payload ICD.

Payload Physical Interface

Need additional hardware engineering done before we understand the physical implementation of the cabling and connectors for the payload connection to OiaB.

Implementation

Implementation decisions and issues concern the architecture considerations for how we realize functionality with software or hardware.

CM File Transfer

The team must research the ability for two Visual Source Safe products to perform file transfers with each other across a network. Issues include security, permissions, routing, domains, trust relationships, subnet specifications, and NT registry interaction. An alternative strategy would be to export the VSS data unit to a blob, then use FTP to move the blob.

Database Scope

The database server and contained tables will be accessible simultaneously by multiple users. The CM functions will prevent multiple-write collisions. The level of granularity on a controlled access (one person with write ownership) depends upon the data unit size. While only one user may have a data unit checked out with a lock (for write access) many users may have a copy of the data unit checked out for read. We also agree that the server and its database tables are in a well-known and static location.

Disk Configuration

The POST Tools PC should have its total disk storage divided onto two disks, C: and D:. The operating system and other high-level services will be on the C: disk, while the user applications and data will be on the D: disk. This will enable wipe and restore capablity on the C: disk without affecting user data. The C: disk should have at least 1 GB to allow for system logs and growth.

Display Builder Container

The team has decided to provide the SEMO portable display builder container as part of the system. While the PDB beans can work in any integrated development environment (container), providing the PDB container ensures support for bean package and simplifies the development task. Customers can elect to use a different IDE.

ISP Logs

Need a study and decision on the name and location of ISP server log files on both the POST Tools PC and the Orbiter-in-a-Box.

ISP SITF on NT

The team needs to determine the availability, services and performance of an ISP SITF server for Windows/NT. Determine feasibility of editing SITF files with Notepad or Wordpad, saving as text files.

InstallAnywhere Loads

Does the version of InstallAnywhere we already have enable us to install images and files other than Java?

InstallShield Loads

Need to understand how InstallShield licensing works. Does it allow us to build one load and distribute it N times, for one fee.

Java Version

We have chosen to baseline Java 2 (JDK 1.2.2, J2SE, J2EE, JAXP 1.0, JH 1.1) for the project. This package provides features important to beans, security, properties, graphics, and performance.

PDB Save Format

PDB will use Java archive (JAR) files to store serialized displays.

PLS Data Stores

Data store save and recovery is file system dependent. A solution will depend upon payload server implementation and the mode and control design for the server and plug-and-play models. A related issue concerns how the orbiter-in-a-box can synchronize an SMS data store with a payload data store.

PLS Implementation

The implementation of the plug-and-play payload server holds some uncertainty. The desired outcome is to reuse the payload server code designed for the SMS installation on the orbiter-in-a-box. There are some OS, human interface, and physical interface compability issues here for the two deployments.

Product Generation

Our initial position is that the system provides a list of canned predefined products from which the user can select. The system should include the ability to support remote additions through plug-in style enhancements.

Report Generation

Our initial position is that the system provides a list of canned predefined reports from which the user can select. The system should include the ability to support remote additions through plug-in style enhancements.

SSP in NGFCT

How is the Standard Switch Panel model implemented in the NGFCT platform? Does it communicate with a payload model, and if so, how? Is this panel included in the set of panels provided? Strategy: if not, need to implement another panel in the panel set.

VxWorks

The orbiter-in-a-box platform will run the GPC emulator, SMS models, and supporting components on the VxWorks operating system. We chose VxWorks over candidates OSE, Lynx, Linux, and pSOS because of the integrated development environment, availability of board drivers, 3rd party tool support, and standards compliance. We also have some experience in developing software for this operating system.

Windecom File Format

The content and structure of the telemetry definition file that the Windecom tool requires is unknown. It is not known how the file is built and where the creation tool resides. It is not known how Windecom will handle orbiter definition and payload definition coming from two different sources.

Windows/NT

We chose the Windows/NT 4.0 operating system over competitors for the POST Tools PC platform because we need server support, high reliability, and security features. This operating system enables the project to deploy virtually any 3rd-party COTS tool for application implementation, and provides a strong API for integrating new software with the operating system and utilities. We will evaluate Windows/2000 for maturity and stability when available. We did not choose Linux because the nature of the application tools we are building are closer to the Windows style and customer familiarity.

Operation

Operation decisions and issues refer to concerns about how we intend to operate the system upon deployment.

Administrator Roles

A POST tools PC account should be configured for a POST field engineer role, giving this user rights to create accounts and perform other system administrative functions without granting all system administrator privileges.

Help System

The help services will be presented in a separate frame, as is consistent with the scenario presented in the Vision document. The separate frame might be another browser window, another browser instance, or a separate help system browsing utility.

PLS Mode and Control

The plug-and-play payload server mode and control function implementation is not well understood by the development team. The desired outcome is a portable GUI (e.g. Java) that the POST tools project can reuse to communicate with the payload server.

Payload Instructor Displays

The customer will write the payload instructor displays with the PDB tool, publishing malfunction and moding instructions to the model via ISP. This is identical to the way the SMS instructor pages work in the NGFCT facility. NOTE: Implies need for PDB ISP publish capability.

Reconfiguration

Reconfiguration decisions and issues concern the architecture considerations for program and data changes in the field.

ISP Dictionary Maintenance

The orbiter-content ISP dictionary will have to be maintained securely at the site. Only authorized users should be able to edit or update this file.

Import Browser

The issue regards how to identify and read foreign files into the import-validating subsystem. One strategy is to use the web browser to invoke the Java file system browser, assuming permissions are set to allow an applet to read/write local file system (preferred). Another alternative strategy is to use the Windows explorer with an OLE API hook into the application.

Install Verification

Can we use Robot to verify a field installation? At issue is (a) whether we can build to deployable kit to use robot in the field, perhaps without a full test suite license, and (b) whether it can see administrative dialogs.

PLS Reconfiguration

The method and content of the reconfiguration task for the payload server is unknown. We expect the content to come in some way from the CDT and a collection of static (fixed) configuration definition files.

Payload GAF

The ispsmserver in the OiaB builds its dictionary from GAF files. At issue is how to build the payload symbol table into a GAF that the ispsmserver can include in its auto-build.

Payload Server Variables

Need to examine the collection of name mapping subroutines being developed for the SMS payload server and determine their applicability to the Orbiter-in-a-Box services for SMT. Of particular interest are the "regvar" services.

Payload Symbol Designators

The issue arose in examining how to constrain the payload training model selection of input/output symbols (was MSID's) according to what's available in the (a) payload sever, (b) ISP server, and (c) CDT reconfiguration data. We don't want to have to maintain separate dictionaries for each tool, because these easily could get out of sync.

The primary strategy is to designate CDT the role of providing the master collection of symbols (MSID's) and create the dictionary that SMT and PDB use as their selection source. CDT already is providing something like this for Cargo PC telemetry definition. This would force the payload customer to use CDT first, to create the master dictionary, but would also simplify other development and testing tools (for example, CDT would collect customer-created symbols for training model malfunction publish terms).

SMT MSID List

The team remains undecided regarding the dynamic creation of the MSID list for the SMS model tool in the field. Part of the MSID list comes from an ISP dictionary for the orbiter, and part of the MSID list comes (preferably) from an ISP dictionary for the payload, created somehow by CDT or related tools. The merger tool itself remains undefined as well.

Requirements

The requirements package contains the abstract functional and quality requirements that drive the architecture design element selection. Some of these we describe in quality scenarios.

Abstract Functional

This package captures the abstract characterization of the functional requirements, together with a characterization of the coarse variability and dependencies within those requirements. We consider particularly the end users.

Cargo PC Commanding Function

The fundamental premise of the development and test support capability is the ability to support Cargo PC software development. To provide test support in the field the POST tools must be able to simulate the GPCF and its MDM SIO command channel.

Cargo PC Telemetry Function

The fundamental premise of the development and test support capability is the ability to support Cargo PC software development. To provide test support in the field the POST tools must be able to generate a telemetry stream for the Cargo PC.

Display Portability

The customer creates a set of monitoring and command displays for his payload. Application of the displays depends upon scenario and platform, but most require the same content features and operability. To minimize the number of displays the POST tools must provide a development and test environment that enables the displays to be used for training, ground operations, launch support, and even in-flight operations.

End-to-End Test

Given the customer's development of command and data table information, the training model, and the Cargo PC application software, the POST tools is meant to support and end-to-end (Cargo PC to payload, or Cargo PC to payload model) testing capability in the field.

Payload Training Model

One of the development activities that the customer performs is the SMS payload training model. The POST tools must support development and test of this model in the field, so that when the customer delivers the model to JSC it will plug-and-play into the SMS platform.

Abstract Quality

The abstract quality requirements capture architecture options with regard to performance, utility, accuracy, and other quality dimensions. In particular, these identify a specific stimulus and the desired response.

Auditing

The POST tools should provide a comprehensive set of auditing functions to ensure the quality of the customer's input.

Field Upgradeability

Give the variety of customers and product consumers, we anticipate not having a complete set of report, product, import or export capabilities in the early releases. The system must be able to accommodate new capability additions for these four functions.

GPCE Performance

The GPC emulator (GPCE) capability provided within the orbiter-in-box must cycle the flight software at an apparent rate of 1:1 (25 Hz for HFE). This also includes cycling the SMS models, PDI, PCMMU, and ISP functions at the expected rate.

GPCF Interface

The orbiter-in-a-box tool must provide accurate emulation and simulation of the orbiter avionics to support Cargo PC command application development and testing. The orbiter-in-box must support the GPC payload command filter (GPCF) and MDM interface card (MIC).

Multi-User CM

To ensure the integrity of the data units, the POST tools must provide concurrent configuration management of data units both in the field and at the home site. The tools must allow only one user to have write authority on a data unit, while many users may

have simultaneous read authority. Attempts to write to a data unit which has not been locked (reserved) by the author will not be permitted.

Training Model Performance

The SMS training model client and server functions must be able to cycle the models at the desired rate of 25 Hz.

Quality Scenarios

Quality scenarios make concrete the quality requirements. These are very specific expansions of the quality requirements.

Activity

An activity supporting the component framework identifies a component to perform the desired task. The framework employs the component through a standardized interface.

Component

Components supporting the standardized component framework interface can be added to the system after deployment, supporting evolution of the platform capabilities.

TOTALS:

59 Logical Packages
283 Classes

LOGICAL PACKAGE STRUCTURE

- Logical View
  - Requirements
    - Abstract Functional
    - Abstract Quality
    - Quality Scenarios
  - Decisions
    - Reconfiguration
    - Implementation
    - Accuracy
    - Operation
  - Constraints
  - Architecture
    - Templates
    - Subsystems
      - Data Collection Services
        - Import Data Services
        - Manual Entry Services
          - Validation Services
          - Editing Services
        - Storage Services
      - Development and Test Services
        - Development Services
          - Training Development Services
          - Display Development Services
        - Test Services
          - Unit Test Services
          - Integrated Test Services
            - Training Model Test Services
              - Payload Server Services
                - SMS Model Services
                - GPCE Model Services
                - Mode and Control Services
                - Model Communication Services
                - Server Reconfiguration Services
            - Cargo PC Test Services
            - Display Integrated Test Services
            - Orbiter Simulation Services
            - Payload Integrated Test Services
        - Reconfiguration Services
      - Data Consumption Services
        - Report Services
          - Production Services
          - Report Consumption Services

- Export Data Services
- Product Services
  - Product Consumption
  - Processing Services

Administration Services
- Remote Services
- Customer Site Services
- USA Site Services

Navigation Services
- Internal Navigation Services
- External Navigation Services
  - Identify File Services

Configuration Management Services

Translation Services
- Validate Structure Services